Cloud Based Malicious PDF Detection Using Machine Learning

Viorel Gurdiş
Department of Computer Science
Babeş-Bolyai University
Cluj-Napoca, Romania
viorelgurdis@gmail.com

Christian Săcărea
Department of Computer Science
Babeş-Bolyai University
Cluj-Napoca, Romania
csacarea@math.ubbcluj.ro

***Abstract*—PDF documents are one of the most popular file formats used for information exchange. People trust this format and open the files as soon as they come into their possession. Few know the risks to which they expose the integrity of their computers. For already many years PDF files became a frequently used attack vector for compromising the security of computers. This paper will introduce you into some of the currently known PDF attack techniques. In addition, our work is aimed to research if the Machine Learning algorithms can successfully reinforce the classical antiviruses at analyzing malicious PDF documents. For this purpose we have chosen the Random Forest classifier, that was trained using features representing possible indicators of presence of malware in the PDF format structure. By integrating the trained model into a Cloud based framework we provide an alternative security solution, which could take the responsibility of the tedious task of malware analysis to an isolated, high-performant environment in the Cloud. To automate the process of detecting and analyzing a PDF file, we created a Windows Service that runs permanently in the background and is registered to the filesystem events. Thus, each time a PDF file is downloaded, the corresponding filesystem event will notify our Windows Service, which will then upload it to the Cloud analyzing API. Detected harmful PDF documents will be removed from the computer and users should not worry about their data privacy nor computer performance. While this paper focuses on researching malware in PDF format, the presented Cloud API is designed to easily integrate other Machine Learning models, trained to detect malware in different file formats.**

***Keywords*—Cybersecurity, Machine Learning, Malicious PDF, Cloud Application**

I. Introduction

The informational technology progress brings a lot of benefits along with new responsibilities. 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 transferred 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 hackings 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.

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 to represent a threat to the network affiliated to an institution. The cyber attackers could succeed in achieving their goals, but 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, dynamic behavior evaluation and other techniques which are effective for detecting previously identified malware. 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 requirements they need. In this article we will go through a new approach that implies transferring complex detection algorithms from user's computers to a remote service, whose only task is analyzing suspicious uploaded PDF documents. Consequently, the computers that will use this solution get rid of additional workload while still maintaining the system secure.

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

- **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.

- **Development of an 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.
- **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 a user friendly dashboard 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 the scanning results. This is a perfect option for users that require the Cloud features from a smartphone.

The remaining parts of this paper are structured in the following manner: Section II contains the research made in the past years on malicious PDFs and Cloud Computing as an antimalware solution. Section III introduces the reader into the PDF format structure, as well as presents types of malware in PDF. In Section IV we present a Proof of Concept for developed classification model based on recreating a pseudo attack using malicious PDF. Section V is putting each component together and presents the overall design of the application. Section VI is meant for our final conclusions and we also show some of the future ideas for improving the application.

II. Related Work

Beginning with the first reported occurrences of malware, the security researchers have made a huge effort to prevent its 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 subsections we will analyze other academic and industrial approaches for transferring antimalware engines to the Cloud and some documented detection techniques of malicious PDFs.

A. 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** [1], 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 approx. 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]. Its 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. Its 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.

B. Detection of malicious PDF

PJScan, presented in the paper of Laskov and Srndic [3], 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. The obtained sequence of tokens 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 [4]. 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.

[1]Adobe Flash file format used for multimedia

A completely new approach mentioned in the research paper of Fettaya et al. [5] 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 preprocessing. 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 approx. 80% of the malware into different categories.

III. Background

A. Analyzing PDF File Structure

In this subsection we are going to understand PDF structure which is a foundation stone of identifying backdoors in this file format.

PDF was first developed by Adobe in the 90s with many versions being 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 [6], 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` (Fig. 1). 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.

B. 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. (Fig. 2)

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.

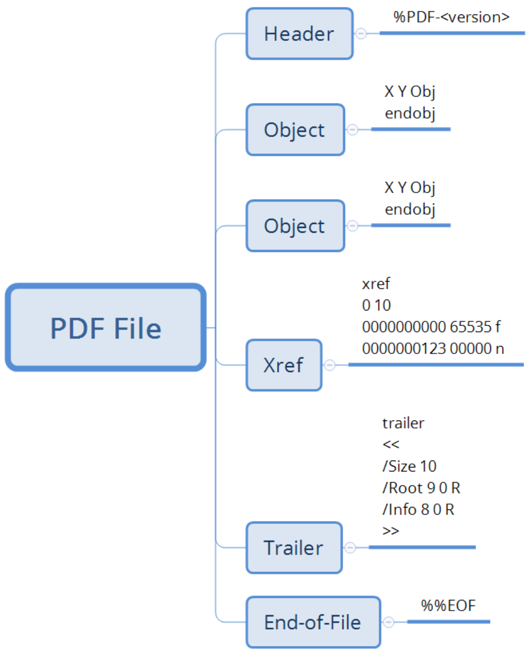

Fig. 1. Structure of PDF Format [7]

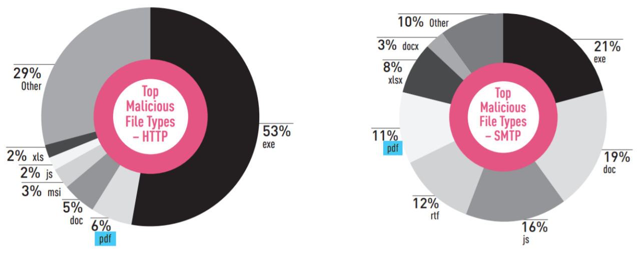

Fig. 2. Top malicious file types - H1 2019 [8]

Before these weaknesses are patched[2], attackers create exploits to execute arbitrary code, when a PDF file is opened with one of the vulnerable applications. Phishing attacks are more difficult to detect, because in the context of PDF documents, they aren't infected, instead 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 [9], it's usually the fault of embedded Javascript code, whose intentions could be wrong, but as seen in Table I 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

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

TABLE I
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

"trick" to complicate the analysis work for both antiviruses and researchers. A PDF document from which it is impossible to copy text, or which cannot be printed, is such an obfuscated sample.

IV. PROPOSED APPROACH

For solving the problem of detecting malicious PDF documents, we came with a Machine Learning based approach. We took into account the importance of keeping the documents content confidentiality, as well as the need for ensuring the speed of analysis process. This solution will be next integrated into a Cloud framework responsible for remote documents scanning.

In the following subsections we will touch upon algorithms and techniques we have used to achieve the purpose of this article, namely detecting malicious PDF documents using a Machine Learning based solution. Fig. 3 represents the entire process which was followed while developing the *ML* model. Each process phase will be fathomed in the following.

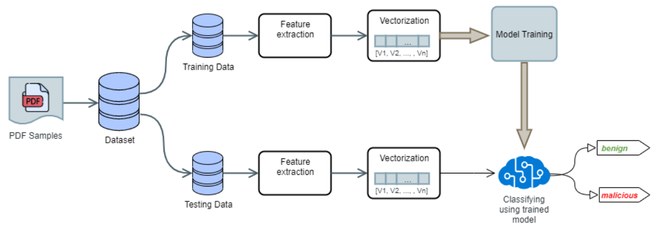

Fig. 3. Machine Learning model creation

A. Dataset

The first step in building the optimal *ML* model was to create a dataset which would train the classifier. Our target was to collect malicious and benign PDF samples used in real life scenarios. Circa 80% of the clean and malicious documents were downloaded from the online malware repository *Contagio* [10], which provides samples collected from various open sources. Nevertheless, for more recent malicious PDFs we used VirusTotal [1], Hybrid Analysis [11] and VirSCAN [12], that allow searching files by their hash (*MD5*, *SHA1*, *SHA256*). Some of them provide access to the entire file collection so that we can order them by upload date and search for the most recent uploaded harmful files. Many clean samples were collected from public sources using Google Search Operators, i.e., `filetype:pdf`, for restricting search results to PDF files. Additionally, we completed the dataset with more than 100 clean interactive PDF files, that contain embedded 3D Widgets, incorporated JavaScript games etc. These clean samples should train the ML model, to correctly classify files even in corner cases.

TABLE II
COLLECTED DATASET OF PDF SAMPLES

Category	Source	Count
`benign`	Contagio, Google Queries, Tetra4D, Pdf-Scripting	9.153
`malicious`	Contagio, VirusTotal, Hybrid Analysis, VirSCAN	10.982

B. Feature Selection

As we already know, Machine Learning is based on mathematical concepts, hence the algorithms consist of performing mathematical operations to identify patterns in data. In order to use our dataset as input for ML algorithms, first of all we need to bring the data in an appropriate form. Therefore, the most relevant data from a PDF document should be extracted and transformed into a vectorized form. As already seen in Table I, there are several standard PDF entries which can be used for malicious intentions. For featurizing PDF documents we have used *PDFiD* from *PDF Tools* suite [4], which is a Python script that selects 22 features from a PDF file, including keywords commonly found in malicious documents, name obfuscations etc. The output of the script is a list of PDF entries and their occurrences. Before turning this featurized form of the PDF file into input for ML algorithms, the data should be normalized to [0, 1] range. This is an important step in order to avoid ML issues when features are on drastically different scales (see [13]). The vectorized form of the PDF document is built by applying the *Min-Max Normalization* (1) strategy on the array of features extracted by *PDFiD*.

$$z = \frac{x - min(x)}{max(x) - min(x)} \quad (1)$$

C. Classification Technique

As we have successfully created the dataset by vectorizing each PDF from our collection and setting a label for each one (everything saved as a CSV file), the next steps would be feeding the dataset to a ML classifier and training a model. According to our application's requirements, a PDF file should pass through a binary classification, so the analyzed files can be only `malicious` or `benign`, respectively a supervised strategy was chosen. The model, that we will obtain when applying Machine Learning, is able to perform tasks that it has not been explicitly implemented to execute. First of all we split the dataset: 70% were used as training data and 30% were allocated as testing data. The training data is used to fit the model, so that the resulting trained model could be used to predict the correct verdict on previously unseen metadata.

Above all existing classification algorithms, **Random Forest** was selected because of its effective classification capability and the ease of use, as stated in [14]. Under the hood, this algorithm gives the classification result based on assembling the results from multiple decision trees. First a random subset from training data is selected. At each split point, m features from n available are randomly selected ($m \leqslant n$) and the optimal split point from m features is picked ($m = \sqrt{n}$ is recommended). This step is repeated until an individual tree is trained and each result represents the vote of a decision tree. These votes are combined to give an unified final classification result of the Random Forest technique.

Even if the training process is computationally expensive, once the classification model is constructed, categorizing is quickly executed. In order to make our API that integrates this model more efficient, we used the Python module - `pickle` to dump the virtual model into a file saved to disk. The process of refitting the model before using it to predict is much more time consuming than just loading the already fitted model form the disk.

D. Experiment

Before putting the chosen classification algorithm into work, we definitely had to take several aspects into consideration. The most important of them is the environment where the model is trained. When working with malicious samples it is crucial to ensure that the personal computer's integrity will not be affected. The instructions defined in Lenny Zeltser's chapter "Analyzing Malicious Software" from [15] helped us to set up an isolated laboratory environment for our *experiment*. First of all we used VMware to create a virtual machine (VM), where we installed a Linux distribution. In this way no physical machine could be damaged and the lightweight OS keeps the performance unimpacted. For further performance evaluations (see Table III), the allocated hardware specifications for the VM should be taken into consideration: *RAM = 3GB*, *CPU = 4 processor cores*. The communication with the development computer is through a shared folder, where the collection of PDFs, as well as the source code for training the model were dropped. After configuring all the required libraries, the network adapter of the VM was completely removed, so that it could not establish any connection to external sources in case of any malicious behavior. Also, a snapshot of the set up environment was taken. This is another huge advantage of a virtual machine: in case of accidental opening of a malicious file, we could easily revert to a safe snapshot.

TABLE III
RUN TIMES ON TRAINING THE MODEL ON DESCRIBED DATASET

Operation	**Time**
Feature Extraction	29 min
Classifier Training	0.405 sec
Classification	0.07 sec

After the classification model was successfully created, we decided to simulate a cyberattack using a malicious PDF file, in order to test the model prediction in a pseudo-real-life scenario. Metasploit[3] was used for injecting a malicious payload into an innocent looking file. After uploading the file to our analyzer, the classification result was as expected - `malicious`.

E. Performance Evaluation

As performance estimators for evaluating the strengths of the obtained classification model we have selected the following metrics, where *TP* stands for True Positives, *FP* - False Positives, *FN* - False Negatives and *TN* - True Negatives:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$F1 - Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5}$$

		Predicted	
		benign	malicious
Actual	benign	TP = 2708	FP = 1
	malicious	FN = 2	TN = 3329

Additionally, we generated a confusion matrix to observe the presence of FP and FN. The results of testing on 6640 samples (30% of the dataset) show an accuracy of **99,95%** for the trained model using *Random Forest Classifier* with 100 estimators. Of course there are some minor corner cases, when the detection fails: 2 files were detected as benign while they were malicious and 1 benign file was predicted as malicious. However, the overall results (see Table IV) demonstrate that the obtained model can be efficiently applied for malicious PDF files detection.

TABLE IV
RANDOM FOREST CLASSIFIER PERFORMANCE RESULTS

Accuracy	**Recall**	**F1-Score**
0.995	0.9992	0.9994

[3]a penetration testing framework

V. Application design

In this section we are going to have an in-depth look at the architecture of our application - **ExtWatcher**, that represents a solution for the enunciated problem in this article.

A. Architecture

ExtWatcher is a software application assembled from multiple components using different technologies (see Fig. 4). In the following we will introduce you into its implementation design starting from the low level and we will explain what role each component plays in the context of providing a truly effective security solution.

The core of the entire software application is the *Cloud Analyzer*, which provides the security measure against malicious PDF documents. ExtWatcher has two different, independent approaches for getting access to the Cloud Analyzer.

- The first one implies automatic file detection and requires the installation of *ExtWatcher Service*, which will establish a connection through HTTP to the Cloud Analyzer and will instantly remove detected malicious files from the user's computer.
- The second approach is an *out-of-the-box* functionality, the user being able to analyze a file by accessing the Cloud Analyzer Dashboard (web graphical user interface) and entering the URL of the file. The dashboard submits the URL to the Cloud Analyzer and the later downloads the file, classifies it and sends back the verdict. This solution is fit for mobile devices, as it requires no additional installation, ExtWatcher Service being currently supported just on Windows OS.

In order to properly work, our application requires permanent Internet connection, because the core of analysis can be remotely located. The advantage of this is the fact that Cloud servers are better at applying Machine Learning models for malware detection, thereby reducing the workload on personal computers.

B. Windows Service

This subsection will describe how the Windows Service manages to simplify the user responsibility and to automate the task of detection and uploading of PDF files to our analysis engine. As previously mentioned, this is not a mandatory component, rather just a help utility for users whose computers work on Windows platform. The Windows Service starts running at the boot-time of the computer in the *Session 0*[4] and communicates with an application responsible for notifying users through Windows System notifications (see [16]). An overview of the described functionality design is presented in Fig. 5). The mentioned features make Windows Services suitable for long-running functionality, whose operation doesn't intersect with other users who work on the same computer.

[4]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 user can be sure, that once he downloads any PDF file to his computer, the integrity of the system is protected by our application. Basically, the Windows Service detects that a new file of type PDF was downloaded, blocks and hides it, so that the user cannot access the file until it is not scanned. Next, the Windows Service submits the file to the remote server for analysis. While waiting for the response, it sends an event to the *Tray Notification Application*, which is responsible for pushing System Notifications to the user, saying that the downloaded PDF is being scanned. After the analysis response in JSON format is caught by Windows Service, it takes the corresponding action on the PDF file, based on the *verdict* extracted from JSON.

- `benign` - this verdict means that the Windows Service can take out the applied restrictions (block and hide) from the file.
- `malicious` - Windows Service should immediately remove the file from the disk.

The implementation design of ExtWatcher Service takes in consideration further evolution of our product. So, it can be easily configured to monitor more file extensions, when the Cloud Analyzer will contain support for analyzing more file formats.

C. Cloud Analyzer

Next we will finally cover the core of our product - the Cloud Analyzer. There are multiple ways to access the core, as it is a Cloud based solution. The first two methods are straightforward and make the use of the Cloud application intuitive. Nevertheless we designed our API keeping in mind the REST architecture. As effect, we have a strong separation between the server and client side, thus giving the opportunity to our API to be integrated in multiple ways. Beside the *visual* method (web Dashboard Interface) and the *automation* method (Windows Service), this framework can be adapted to more use-cases. The custom client can access it via the exposed REST interface, no limitations being currently applied. This independence of the Cloud Analyzer assured by the RESTful design makes it flexible and portable, permitting to configure it in different types of environments. It was not the purpose of this paper to deploy the framework to any Cloud service, but it was a main goal to achieve its portability and ease of configuration.

In order to retrieve information about a PDF file, the *ExtWatcher Service* uses a `POST` HTTP method to upload the file to the framework. Once the PDF is uploaded to the server, first of all, the MD5 hash of the file is calculated to check in the database if the same file was not already analyzed. If the hash is not found, the file is being analyzed by applying the ML model for predicting the file maliciousness. The result together with a HTTP status code is then sent back as response.

D. Dashboard Interface

ExtWatcher also offers an intelligent solution for interacting with the Cloud Analyzer API and to visualize the data that passed through it. This is a responsive web Dashboard with

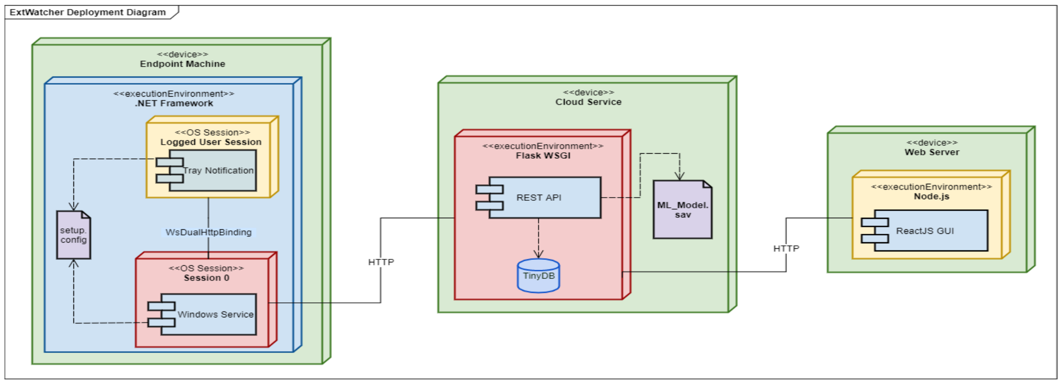

Fig. 4. Deployment Diagram of the application and used technologies

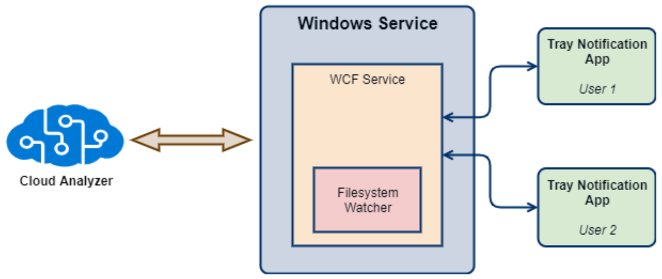

Fig. 5. Architecture's outline of the help tool for Windows

the help of which users can interact with the API from any device, be it laptop or smartphone.

It was implemented separately from the server, so it calls just the API methods it requires, without being forced to implement a concrete server interface. At the moment, any user that accesses the Dashboard has the possibility to view statistics about all files ever scanned, including file details such as: filetype, MD5 hash of the file, filename, date when it was scanned and result of analysis.

Also the dashboard presents a "feed", that serves as a history of using the Cloud Analyzer. It includes information about how each file came to be analyzed: by being uploaded by the ExtWatcher Service or by manually submitting the URL of the file, as well as information about the IP the file was originated from.

To simplify the interaction with this web application, the app provides also a *searchbar*. The user can quickly find information about any file, by specifying a basic search query.

For those users who adopt the automatic detection and submission to the Cloud Analyzer approach, using the *ExtWatcher Service*, the Windows Installer can be downloaded directly from the Dashboard. Users that don't prefer this method or simply don't have a computer with Windows OS, could still analyze PDF files directly from the web Dashboard with the previously mentioned method. The user should just specify the URL of the file he wants to analyze and the rest of the workload is taken by the Cloud Analyzer. In this way, the user can be confident that no PDF file can harm his device, because it is first downloaded by the Cloud Analyzer and is analyzed on a remote, safe environment. After the analysis result is showed, the user can decide what to do with that file.

VI. Conclusion and Future Work

The main objectives behind the proposed approach for creating a Cloud application, which could scan PDF documents and return a verdict was successfully fulfilled. This paper demonstrates the capabilities of Machine Learning algorithms applied on cybersecurity tasks, which in some cases can provide even better results than a classic antivirus application. In addition to the Machine Learning framework responsible for PDF scanning, we also provide an automated solution for file uploading, which can take an according action to the scanned file based on its result, in order to assure the safety of user's computer. The symbiosis of these two creates a complex system that distributes the performance-expensive task of malware analysis from an endpoint machine to a remote server. This way the user should not worry that his computer will be compromised by any PDF document, which these days is one of the most exchanged file formats.

The next step in extending our created product will be making the framework publicly available, by deploying it on a Cloud service. Currently the automatic file uploader is supported only on Windows OS and the users have the alternative to manually submit the URL of a PDF file in order to scan it. In the future we plan to develop a similar background application for automating file upload for Android OS and possibly other operating systems. The Cloud application is also included in our upgrades plan. Its architecture already permits the integration of other Machine Learning models for classification of new file formats. The purpose of the upcoming research will be Microsoft Office Document formats (DOC, XLS, PPT), as well as EXE (Executable file) and DLL (Dynamic-link library) files.

We think that now is the era for transferring all of the performance draining applications to the Cloud.

References

[1] "The VirusTotal website." [Online]. Available: https://www.virustotal.com/

[2] Bitdefender, "Sandbox Analyzer." [Online]. Available: https://download.bitdefender.com/resources/files/News/CaseStudies/study/211/Bitdefender-2017-TechnicalBrief-SandBoxAnalyzer-crea2103-A4-en-EN-2-GenericUse.pdf

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

[4] D. Stevens, "PDF Tools," 2008. [Online]. Available: https://blog.didierstevens.com/programs/pdf-tools/

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

[6] A. S. Incorporated, *PDF Reference, sixth edition: Adobe Portable Document Format Version 1.7*. Adobe, 2006.

[7] LogRhythm, "Detecting Potentially Malicious Javascript Embedded Within a PDF File Using LogRhythm Netmon." [Online]. Available: https://logrhythm.com/blog/detecting-malicious-javascript-in-a-pdf/

[8] C. P. Research, "Cyber Attack Trends: 2019 Mid-Year Report." [Online]. Available: https://www.checkpoint.com/downloads/resources/cyber-attack-trends-mid-year-report-2019.pdf

[9] X. Magazine, "Looking for exploits in pdf-documents on our own," 2014. [Online]. Available: https://xakep.ru/2014/09/26/search-document-exploit/

[10] M. Parkour, "Contagio malware dump. Malicious files for signature testing and research." [Online]. Available: http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html

[11] "The Hybrid Analysis website." [Online]. Available: https://www.hybrid-analysis.com/

[12] "The VirSCAN website." [Online]. Available: https://v.virscan.org/

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

[14] C. Chio and D. Freeman, *Machine Learning and Security*. O'Reilly Media, Inc., 2018.

[15] J. Bayuk, *CyberForensics: Understanding Information Security Investigations*. Humana Press, 2010.

[16] P. Yosifovich, A. Ionescu, M. E. Russinovich, and D. A. Solomon, *Windows Internals*. Microsoft Press, 2017.