Identifying Drawbacks in Malicious PDF Detectors

Ahmed Falah(✉), Lei Pan, Mohamed Abdelrazek, and Robin Doss

School of Information Technology, Deakin University, Burwood, VIC 3125, Australia
{afalah,l.pan,mohamed.abdelrazek,robin.doss}@deakin.edu.au
https://www.deakin.edu.au

Abstract. Despite the continuous countermeasuring efforts, embedding malware in PDF documents and using it as a malware distribution mechanism is still a threat. This is due to its popularity as a document exchange format, the lack of user awareness of its dangers, as well as its ability to carry and execute malware. Several malicious PDF detection tools have been proposed by the academic community to address the PDF threat. All of which suffer some drawbacks that limit its utility. In this paper, we present the drawbacks of the current state of the art malicious PDF detectors. This was achieved by undertaking a survey of all recent malicious PDF detectors, followed by a comparative evaluation of the available tools. Our results show that Concept drifts is major drawback to the detectors, despite the fact that many detectors use machine learning approaches.

Keywords: Malicious PDF detection · Comparative evaluation
Concept drift

1 Introduction

It has been getting increasingly popular to embed malware in documents. PDF in particular, which is used as an alternate distribution mechanism. This is to counter users' increasing awareness of the dangers of executables and other malware distribution approaches. In comparison, not as many are aware of the capabilities of PDF (and other *seemingly benign file types*) and its ability to carry malicious code. Users are blindly assuming it as a plain document, especially when combined with a little social engineering, to trick the target into ignoring any warning signs that might be detected unconsciously. A security-related psychology study in [14] found that for non-security professionals, the human brain is more likely to pick up (cyber) danger indicators unconsciously than it can consciously.

Figure 1 shows a trend regarding the number of PDF-related reported vulnerabilities (as common vulnerability exposures (CVEs)) each year. 2009 seemed to be the peak for maldocs and exploitable documents, given the high number of reported CVEs. There was a quiet period between 2009 and 2016. But more

R. Doss et al. (Eds.): FNSS 2018, CCIS 878, pp. 128–139, 2018.
https://doi.org/10.1007/978-3-319-94421-0_10

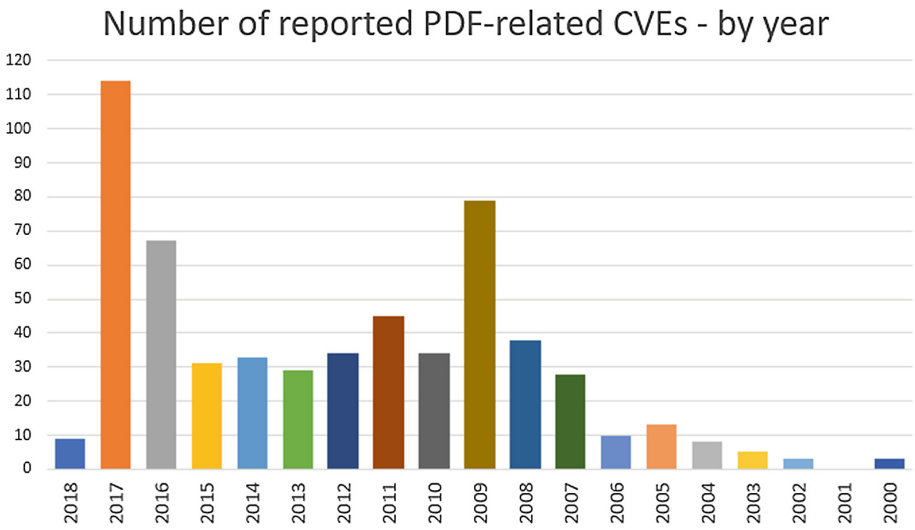

Fig. 1. Number of reported PDF-related CVEs each year. 2016 and 2017 saw an increase in reported CVEs, compared to previous years. Figures collected from [5]

than doubling the number of reported CVEs (31 in 2015, 67 in 2016), which continued in 2017, increasing by nearly 60% with 114 reported CVEs and 8 reported already in the first 2 weeks of 2018. Not only the number of reported CVEs is increasing, but also the severity of these CVEs is increasing, as shown in Fig. 2, according to the NVD [15], where the number of "high" severity increased by over 50% from 2016 to 2017. Figure 2 contains data taken in the last 2 years using the CVSS version 3.0 score. Data of the previous years is still in version 2.0 format. Table 1 shows the NVD scores range for each severity rank.

Besides user awareness, PDF is widely used in business, making it an ideal malware distribution mechanism, because it works across platforms, devices and operating systems, in particular, its ability to execute a wide variety of code, such as JavaScript and ActionScript.

In the September 2017 threat report [12], McAfee stated that malware writers and cyber criminals are moving away from binary and executable into non-executable, script-based malware. This is due to its advantages over

Table 1. NVD's CVE severity score range (CVSS version 3.0)

Ranking	Score
Low	0.1–3.9
Medium	4.0–6.9
High	7.0–8.9
Critical	9.0–10.0

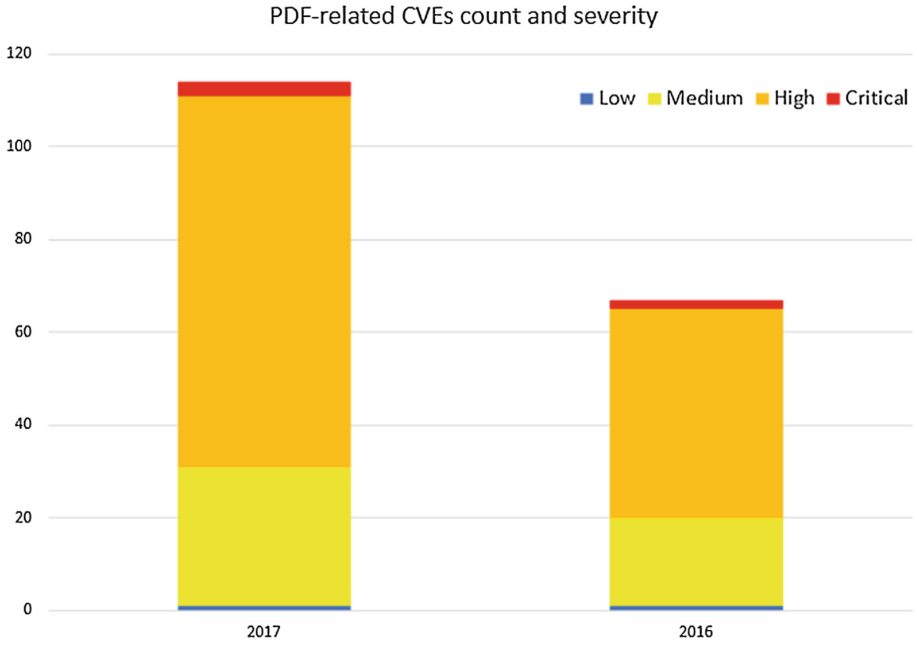

Fig. 2. Severity of PDF-related CVEs in 2016 and 2017. The count of CVEs rated by the NVD as "high" has more than doubled in 2017, compared to 2016. Data obtained from [15]

executables, such as: (1) ease of antivirus evasion, (2) efficiency, (3) easier obfuscation.
Table 2 highlights the current techniques in malicious PDF detection:

1. Static based detection approaches are more utilized than dynamic approaches, specifically, it is used in 4 out of 5 tools reviewed in this work. This focuses on detecting malicious indicators in document structure and metadata, or content (i.e malicious JavaScript). However, there exists a wide range of techniques, such as classifier evasion, parser confusion attacks, to counter static based approaches.
2. Machine learning is used with static based detection (3 out of 4 static based approaches are machine-learning based), and is not utilized in dynamic-based detection approaches.

This is counter intuitive, seeing that recent PDF standard improvements have limited the malicious capabilities of PDF, which leads into the primary installer/dropper role (according to [12]). In this case, a malicious PDF includes a script which can be automatically triggered to download another malware. Such behavior is detected more accurately by adopting dynamic approaches.

Another unexplored area in PDF detection is its utilization in social engineering phishing, where a PDF file does not contain any malicious contents, instead, a file presents a malicious URL to the victim, as well as some message to motivate the victim to click the link. That link leads to a malicious web page that performs a malicious act (such as download malware or steal credentials). In this scenario, a PDF file plays a vital role in the compromise, without actually including any malicious contents.

Furthermore, the current detection approaches are all almost always client- or web-based, where a user is required to manually submit a PDF file for inspection. Submitting a file for inspection requires a considerable amount of effort, and potentially advanced computer skills. This laborous process prevents these tools to be used widely by ordinary users. Situations become worse when there are many files to be scanned regularly. This is the expected case in business environments, where users expect (and need) the security, without the high interaction overhead.

We conducted a set of experiments to identify drawbacks in malicious PDF detection tool. 2 tools where trained with 2 datasets collected over different periods. The first dataset was collected before 2013 (taken from [4]), and the other collected recently and provided by VirusTotal. Our experiments show that concept drifts is a key challenge, where tools trained with data collected in previous years, did not accurately detect the testing dataset. Where detection accuracy decreased from around the 90–100 percentile to the 70–80 percentile. This happened because the testing dataset collected in recent years contains PDF file of other standards, such as the PDF/A standard.

To summarize, the previous section highlighted the following research gaps in the field of malicious PDF detection:

1. The current malicious PDF detection suffer from concept drift and other factors that decrease the detection efficiency and reliability, such as being limited to a specific PDF version or standard of inspected files.
2. The current tools operate at client-level only, and require considerable effort to submit a file for inspection.
3. The tool designers do not consider expected change in PDF distribution mechanism, such as IoT devices (i.e. smart meters) automatically generating reports in PDF (among other formats) and sending them directly to recipients, without going through conventional distribution methods, such as email.

Our contributions in this paper are:

- Conducted a comparative evaluation to identify concept drifts in current state of the art malicious PDF detectors, and other factors that cause drop in performance.
- Identified drawbacks in current state of the art malicious PDF detectors.

2 Literature Review

Between 2000 and 2017, 572[1] PDF-related vulnerabilities were published on the CVE database [5], 114 of which were reported in 2017 alone, and 67 in 2016, that is 31% of the total vulnerabilities in 2 years only. Despite the several malicious PDF detection methods that were proposed dating as back as 2007 spanning over the past 10 years [7–11,16,18,20,22], embedding malicious code within PDF files is becoming increasingly popular among cyber criminals. This is because of its versatility, portability, and wide spread, as well as supporting features that allow malicious code execution. This section will briefly review the features that enable malicious code embedding within PDF documents, then the most recent detection methods will be discussed, and will conclude by discussing social engineering and the role it plays in parallel with malicious PDF.

Enablers: Reviewing the PDF standard [2] shows the rich content allowed in the files, which is part of the reason PDF has become the de-facto file exchange standard in enterprises. The main enabler according to [3,10] is the ability to embed JavaScript within PDF files to perform various tasks, which is notorious for its exploitability, such as these shown at BlackHat [6]. Besides JavaScript, [10] also lists ActionScript as a tool.

Moving away from technical enablers, malicious PDF writers exploit the benign appearance of PDF files. The malicious potential of PDF files is known in the security communities, but non-expert, average users are not aware. Thus the probability of opening a malicious PDF file by a person is much higher than opening files in other formats such executables, regardless of the distribution mechanism (email attachment, USB stick, download) and the presence of anti-malware applications.

Behavior: It is possible to perform sophisticated attacks through PDF, according to [9], such as heap spraying, mapped memory search and DLL injection. [1] shows the reader application is frequently updated and patched, and exploitable embedding formats are blacklisted. These security updates are driving PDF utilization into one of the following malicious roles: (1) The dropper role where the PDF file will download and install a malware from the Internet. (2) Leverage in social engineering attacks, such as including a malicious URL and tricking the user into clicking it, in a phishing-like approach.

Detection: Table 2 summarizes the detection tools reviewed in this section. The table shows that the static approach is more preferred than dynamic approaches. This is because of its speed, efficiency, and low overhead. In comparison, dynamic approaches are slower and more expensive, but could be more accurate. The table also shows that machine learning is utilized in static approaches only, but dynamic approaches do not rely on machine learning detection. Figure 3

[1] Search was conducted using the "PDF" keyword only. [22] reports much higher numbers using assumably the "adobe acrobat reader" keyword.

Table 2. Summary of the reviewed maldocs detectors.

Tool	Year	Method	Focus	ML
PDFrate [16]	2012	Static	Structure[a]	Yes
Unnamed [9]	2014	Both	JavaScript	No
Slayer [10]	2015	Static	Content & structure	Yes
Hidost [19]	2016	Static	Structure	Yes
PlatPal [22]	2017	Dynamic	Behaviour differences	No

[a]Document structure and document metadata are used interchangeably by various works

provides a high level overview of all operations that take place in malicious PDF detection. When a static approach is taken, a detection tool will look at the various tags used in a file, then a classifier makes a decision. Some tools parse JavaScript and review the content of each tag, rather than making a decision based on the tags only. When we take a dynamic approach, the behavior of a file is monitored before a decision is made. Below is a review of the most recent academic detectors.

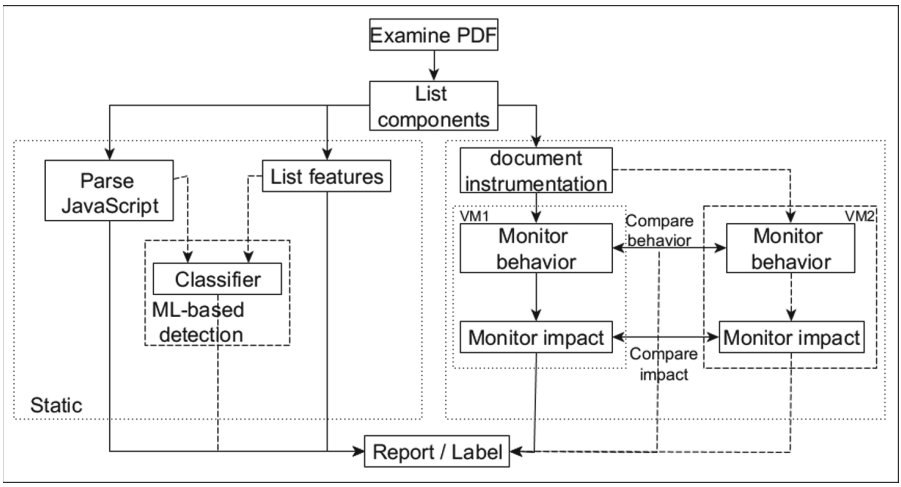

Fig. 3. High level overview of the malicious PDF detection process.

PDFrate [16] examines over 200 features extracted from document structure and meta data and utilizes random forests to binary classify PDF files. It then classifies malicious documents as opportunistic (relies on mass distribution) or targeted (targets specific individuals or organization, utilizing social engineering to lure the victim into interacting with the document). To counter mimicry attacks, the authors suggest removing the top feature that enable such attacks,

resulting in negligible classification errors, a problem that was later addressed in [17]. [18] takes a similar approach to [16], basing their detection on differences in the structure between malicious and benign files. The work was later on improved in [19], where the classifier examines the logical structure and the file content, as well as extending it to cover multiple hierarchical file formats, such as Adobe Flash.

Differently, by focusing on content instead of structure, [9] proposes a content-aware detection approach. Utilizing document instrumentation to monitor JavaScript execution at run-time for certain behaviors such as malware dropping, suspicious memory consumption, suspicious network access, and process creation. The evaluation dataset used provide insight on the trends on malware writers, where every single malicious file out of the 7370 used, contains JavaScript, which justifies focusing exclusively on detecting malicious JavaScript behavior in the work.

[10] highlights the drawbacks of the previous works [9,16], where a structure only detection approach is susceptible to manipulation and mimicry attacks, and a JavaScript only detection is incapable of detecting any non-JavaScript malicious content. To address this, the authors build upon the previous two works, proposing a system that extracts both content- and structure-based information in order to build a classifier that leverages adaptive boosting decision trees and over 100 features. The authors reported resilience of their classifier against three types of attacks: JavaScript injection, EXE- and PDF embedding, making no mention of resilience to mimicry attacks.

Detectors that rely on JavaScript extraction are vulnerable to a new class of attacks introduced by [3], called Parser confusion attack. The attack exploits the weaknesses of the current JavaScript parser, which includes implementation bugs, designers errors, omissions and ambiguities. The attack attempts to hide the malicious payload embedded within a PDF file by encoding and obfuscating the objects, malicious JavaScript and reference. The authors suggest three mitigation technique for the proposed attack: (1) exploit detection at runtime (2) improving JavaScript parsers, and (3) deployment of the proposed reference extractor. Besides these mitigation suggestions, [22] goes a different direction to address this attack class, as well as other techniques. The authors propose a detection technique based on platform diversity. They assume that benign PDF files will behave similarly on different platforms, while malicious files, especially targeted attacks, are designed to attack a specific platform, therefore showing several behavioral differences when examined on non-targeted platform. This behavior difference stems from differences in how various platform handle various aspects, including (1) system call semantics (2) calling convention and argument passing (3) library dependencies (4) memory layout (5) heap management (6) executable formant (7) filesystem semantics (8) expected programs on the target platform, according to [22]. Exploiting specific vulnerabilities requires tailor-made input that utilizes some or all of the factors listed above, which leads to failed, or behaviorally different, execution on various platform.

3 Experiment

The aim of this experiment is to identify concept drift that occurs due to the aging of classifiers. This occurs when new malware samples utilize new techniques that were not examined during the training of the classifiers, resulting in questionable outputs and unreliable classification.

The experiment evaluated the performance of 2 of the tools explained in Sect. 2: PDFrate and Slayer. Both employ static detection approaches, utilization machine learning.

3.1 Datasets

In preparation for the experiment, malicious and benign PDF files were collected from the following sources:

1. Contagio: a large, publicly available dataset that dates back to 2013, among the rest, this dataset contains 9000 benign PDF files, and over 10,000 malicious PDFs. Used for training and evaluation.
2. VirusTotal: provided 10,500 malicious PDF files. Used for training and evaluation.
3. TPN: contains 1000 open-source PDF files. Used mainly for training.
4. Personal: this dataset was used for evaluation, and was collected from personal files, as well as Google searches.

To perform the experiment, the datasets explained above were divided into several sub-datasets for training and evaluation. Table 3 summarizes the datasets used.

Training: 900 benign and 900 malicious files were used from several dataset for training each instance of Slayer.

Evaluation: 10×100 benign and malicious files were used from several datasets to evaluate Slayer and PDFrate.

Table 3. Datasets used in the pilot experiment.

#	Source	Label	Status	Files	Purpose
1	Contagio	Benign	Old	900	Training
2	Contagio	Malicious	Old	900	
3	TPN	Benign	Recent	900	
4	VirusTotal	Malicious	Recent	900	
5	Contagio	Benign	Old	1000	Evaluation
6	Contagio	Malicious	Old	1000	
7	TPN	Benign	Recent	100	
8	VirusTotal	Malicious	Recent	1000	
9	Personal	Benign	Recent	900	

3.2 Procedure

To identify concept drift, 2 instances of slayer were trained. The first one was trained with recently collected data (called Slayer2017 henceforth). The second was trained with an old dataset, called slayer2013. Each of these classifiers was trained as follows:

- **Slayer2017:** Trained with 900 benign and 900 malicious files from the VirusTotal, TPN and Personal datasets.
- **Slayer2013:** Trained with 900 benign and malicious files from the Contagio dataset.

PDFrate does not need training as it is an online tool, and utilize 3 classifiers, trained with several datasets, according to the author and creator, as follows:

- Classifier trained with the Contagio dataset, called PDFrate (Contagio) in this experiment, trained with 10,000 benign and malicious files.
- Classifier trained with data collected from the network of the George Mason university, trained with 100,000 benign and malicious files, called PDF (GMU).
- Community classifier: trained with files submitted to the PDFrate service and is retrained frequently, called PDFrate (community).

Each instance of the tools was evaluated with evaluation datasets explained in Table 3 (both old and recent files). The experiment was repeated 10 times, each iteration used 100 benign files and 100 malicious files.

3.3 Results

A pilot experiment was conducted, using a fraction of the available datasets, summarized in Table 4. Further more, only 2 tools were tested: PDFratefrom [16] and Slayer [10], as they are both currently available.

Table 5 shows a summary of the results for all tools examined. Both instances of Slayer showed high detection accuracy when evaluating 2013 files (old dataset). All PDFrate classifiers achieved near perfect detection accuracy when testing the 2013 dataset. When evaluating a more recent dataset, all classifiers showed decreased performance, where the detection accuracy dropped to 74%–77%.

Table 4. Available datasets.

Name	# of files	Label	Source
Contagio	9,000	Benign	[4]
Contagio	10,000+	Malicious	[4]
VirusTotal	10,500	Malicious	[21]
TPN	1000	Benign	[13]
Personal	900	Benign	Personal & search

Table 5. Summary of the experiment's results. Both tools performed accurately when evaluating the old dataset (˜2013), but performance dropped significantly when evaluating a dataset collected more recently (˜2017).

Tool	Evlautation datase	Accuracy	Missclass. rate	FP rate	Percision
Slayer2017	**2013**	90%	10%	19%	99%
	2017	74%	24%	45%	93%
Slayer2013	**2013**	93%	6%	12%	98%
	2017	78%	21%	33%	89%
PDFrate(contagio)	**2013**	100%	0%	0%	100%
	2017	77%	23%	35%	88%
PDFrate(GMU)	**2013**	100%	0%	0%	100%
	2017	77%	23%	40%	93%
PDFrate(Community)	**2013**	100%	0%	0%	100%
	2017	76%	24%	39%	90%

4 Discussions

Results shown in Table 5 provide insight on the nature of PDF detection. As malware writers started adapting new and improved techniques to embed their malicious code, obfuscate it, or evade detection, classifiers were not able to match such improvements. This could be the result of old training datasets, resulting in aging classifiers, or, the feature sets utilized are no longer relevant.

The PDF standard is continuously improved, introducing new features, and limiting access to older (specifically; more exploitable, dangerous) features. Therefore, the feature set examined by a specific tool must also be frequently revisited, to introduce new relevant features, and exclude irrelevant features. Otherwise, the classifier will suffer from overfitting, or worse, being built for a particular version of the PDF standard, limiting its benefits significantly.

To prove the above point, a number of PDF/A files were included in the 2017 benign evaluation dataset, which make around 40% of that dataset. This change caused all classifiers to perform severely unreliably, as shown in Table 6.

Table 6. Detection accuracy of 2017 benign dataset, which contains around 40% PDF/A files.

	Slayer 2017	Slayer 2013	PDFrate (contagio)	PDFrate (GMU)	PDFrate (community)
Accuracy	54.30%	66.60%	65%	60.13%	61%

Slayer 2013 achieved similar results to PDFrate (Contagio), where both tools were trained using the same dataset (Contagio), with the difference being that

Slayer was trained with 1800 total files, when PDFrate was trained with a total of 10,000 files. Several factors could lead to this, including: (1) Slayer covers all aspects of static analysis, where it examines content, structure, and parses JavaScript, producing features that cover more aspects, compared to PDFrate that considers structure and metadata only[2]. (2) The algorithm used in the classifier: Slayer employs an adaptive boosting algorithm, while PDFrate uses a bagging algorithm: Random Forests.

5 Conclusion

Several tools have been proposed since the appearance of malware-embedded-documents. Specialized tools started to appear around 2012, utilizing different techniques and approaches, ranging from static to dynamic, tools that barely look at metadata, to more advanced that extract and examine JavaScript and other content types, to those that perform real-time monitoring. Despite that, no tool is yet to offer a 1-package solution that counters malicious PDF file, and each tools is susceptible one or more attack type, such as obfuscation, parser confusion, and evasion.

In this paper, we attempted to identify drawback in current state of the art malicious PDF detectors. This was done via a survey of the tools, followed by a comparative evaluation of the available tools. The experiment attempted to identify concept drift in 2 classifiers: PDFrate and Slayer. It was found that the classification accuracy significantly dropped when trained with old dataset, and encountered newer samples collected in later years. The accuracy also significantly dropped when using different PDF formats and standards, such as PDF/A, where the classifiers frequently miss-classified around 50% of benign samples as malicious. Other findings of this paper include the following drawbacks of malicious PDF detectors: (1) Current tools work at the client-level only (user machines), and do not consider the distribution mechanism. (2) Current tools require high level of user interaction in order to submit a file for evaluation.

Acknowledgements. We would like to thank Mustafa Al-Saegh for helping with dataset cleaning and preparation. We would also like to thank VirusTotal, the owner of the Contagio dataset and TPN for providing access to their files. Finally, we are grateful to the authors and creators of PDFrate and Slayer, for providing access to their tools.

References

1. Adobe: Adobe reader security patches (2017). https://helpx.adobe.com/security/products/reader.html
2. Adobe: PDF technology center (2017). http://www.adobe.com/devnet/pdf.html

[2] Neither article Slayer [10] and PDFrate [16] covers the feature set in full details, thus preventing full analysis of the observation.

3. Carmony, C., Hu, X., Yin, H., Bhaskar, A.V., Zhang, M.: Extract me if you can: abusing PDF parsers in malware detectors, In: NDSS (2016)
4. Contagio: Contagio malware dump (2017). http://contagiodump.blogspot.com.au
5. CVE: PDF-related vulnerabilities (2017). https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=PDF
6. Esparza, J.M.: PDF attack - a journey from the exploit kit to the shellcode (2014). https://www.blackhat.com/docs/eu-14/materials/eu-14-Esparza-PDF-Attack-A-Journey-From-The-Exploit-Kit-To-The-Shellcode.pdf
7. Laskov, P., Šrndić, N.: Static detection of malicious JavaScript-bearing PDF documents. In: Proceedings of the 27th Annual Computer Security Applications Conference, pp. 373–382. ACM (2011)
8. Li, W.-J., Stolfo, S., Stavrou, A., Androulaki, E., Keromytis, A.D.: A study of malcode-bearing documents. In: M. Hämmerli, B., Sommer, R. (eds.) DIMVA 2007. LNCS, vol. 4579, pp. 231–250. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73614-1_14
9. Liu, D., Wang, H., Stavrou, A.: Detecting malicious JavaScript in PDF through document instrumentation. In: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 100–111. IEEE (2014)
10. Maiorca, D., Ariu, D., Corona, I., Giacinto, G.: A structural and content-based approach for a precise and robust detection of malicious PDF files. In: 2015 International Conference on Information Systems Security and Privacy (ICISSP), pp. 27–36. IEEE (2015)
11. Maiorca, D., Giacinto, G., Corona, I.: A pattern recognition system for malicious PDF files detection. In: Perner, P. (ed.) MLDM 2012. LNCS (LNAI), vol. 7376, pp. 510–524. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31537-4_40
12. McAfee: Mcafee september 2017 threat report (2017). https://www.mcafee.com/au/resources/reports/rp-quarterly-threats-sept-2017.pdf
13. Trent Nelson: PDF collection (2017). https://github.com/tpn/pdfs
14. Neupane, A., Saxena, N., Maximo, J.O., Kana, R.: Neural markers of cybersecurity: an fMRI study of phishing and malware warnings. IEEE Trans. Inf. Forensics Secur. **11**(9), 1970–1983 (2016). https://doi.org/10.1109/TIFS.2016.2566265
15. NIST: National vulnerable database (2017). https://nvd.nist.gov
16. Smutz, C., Stavrou, A.: Malicious PDF detection using metadata and structural features. In: Proceedings Of The 28th Annual Computer Security Applications Conference, pp. 239–248. ACM (2012)
17. Smutz, C., Stavrou, A.: When a tree falls: using diversity in ensemble classifiers to identify evasion in malware detectors. In: NDSS (2016)
18. Šrndić, N., Laskov, P.: Detection of malicious PDF files based on hierarchical document structure. In: Proceedings of the 20th Annual Network and Distributed System Security Symposium (2013)
19. Šrndić, N., Laskov, P.: Hidost: a static machine-learning-based detector of malicious files. EURASIP J. Inf. Secur. **2016**(1), 22 (2016)
20. Tabish, S.M., Shafiq, M.Z., Farooq, M.: Malware detection using statistical analysis of byte-level file content. In: Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics, pp. 23–31. ACM (2009)
21. VirusTotal: Virustotal (2017). https://www.virustotal.com
22. Xu, M., Kim, T.: PlatPal: detecting malicious documents with platform diversity. In: USENIX Security Symposium (2017)