1. **Related Work**

Existing maldoc detection methods can be classified broadly into two categories: Dynamic and static analysis. Dynamic analysis, in which malicious documents are executed and examined in a specially instrumented environment in order to capture the samples’ malicious behavior; While for static analysis, the detection is carried out without code execution but by statically scanning and examining the header, binary level N-gram of files etc. In general, the advantages of static analysis are: easy to deploy, good speed but relatively low accuracy. Dynamic analysis, on the other hand, although suffering from low-speed, intense resources-consuming, they enjoy the highest accuracy. Both techniques nowadays already had a large amount of successful stories. More advanced solutions in this line usually involve the hybrids of dynamic and static (Please see Maiorca et al. [9] for detail). A summary of existing methods is presented in Table1.

Table 1: A taxonomy of malicious PDF document techniques. This taxonomy is partially based on Platform Diversity [8] with the addition of works after 2016 as well as summaries parser, machine learning, and pattern dependencies

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Category | Focus | Detection | Work | Year | External Parser? | ML? | Discrepancy? |
| Static | JavaScript | Lexical Analysis [5] | PJScan | 2011 | Y | Y | Y |
| JavaScript | Token Clustering [12] | Vatamanu et al. | 2012 | Y | Y | Y |
| JavaScript | API Reference Classification [7] | Lux0r | 2014 | Y | Y | Y |
| JavaScript | Shellcode and opcode sig [13] | MPScan | 2013 | N | N | N |
| Metadata | Linearized object path [11] | PDF Malware Slayer | 2012 | Y | Y | Y |
| Metadata | Hierarchical Structure [8, 1] | Srndic et al. | 2013 | Y | Y | Y |
| Metadata | Content Meta-features [24] | PDFrate | 2012 | Y | Y | Y |
| Both | Many Heuristics Combined [8] | Maiorca et al. | 2015 | Y | Y | Y |
| Both | Many Heuristics Combined [9] | Maiorca et al. | 2016 | Y | Y | Y |
| Dynamic | JavaScript | Shellcode and opcode sig [15] | MDScan | 2011 | Y | N | N |
| JavaScript | Known Attack Patterns [16] | PDF Scrutinizer | 2012 | Y | N | N |
| JavaScript | Memory Access Patterns [17] | ShellOS | 2011 | Y | N | Y |
| JavaScript | Common Maldoc Behaviors [18] | Liu et al. | 2014 | N | N | Y |
| JavaScript | Platform Independent Tap Point Identification [2] | tap point | 2016 | N | N | Y |
| Memory | Violation of Invariants [19] | CWXDetector | 2012 | N | N | N |
|  | OS | Platform Diversity [8] | PlatPal | 2017 | Y | N | Y |

From Table1, we can conclude that the main focus of static analysis is JavaScript or Metadata from files. Typical detection technique includes Shellcode and Opcode Sig based MPScan[13], Structure and Content based classification[9]. On the other hand, dynamic analysis is mainly focus on extracting the JavaScript Snippet from file and run them directly in order for malicious behavior detection. Typical work in this line includes behavior based analysis[?] and platform diversity based analysis[?].

We can also see from Table1 that all but three methods use either open-sourced or their home-grown parsers and assume their capability. However, Carmony et al.[20] shows that these parsers are typically incomplete and have oversimplified assumptions in regard to where JavaScript can be embedded. This leads to one of the most important research questions: Whether the external parser is robust? This is because the design and implementation of this kind of external parser is usually simple and not been designed to be secured, in this case, only little effect is needed for the successful evasion of malicious malwares. We call this kind of attack ‘Parser Confusion Attacks’ according to Carmony et. al.[20].

Also from Table1, an important conclusion can be drawn: Machine learning (ML), in general, is NOT fitted for dynamic but for static analysis, which we have not seen a dynamic paper with machine learning but ML has been the ‘default standard’ for nearly all static papers because of their ability in classification/clustering without prior knowledge of the pattern. Typical machine learning work here includes PDFrate[24] and PDF Malware Slayer[11] etc. Nearly all their work claim that their classifiers can attain high accuracy under resource intensive environment, but seldom of them mention the security of their deployed ML models, no need to say a comprehensive study on adversarial machine learning. This raise serious doubts about the effectiveness of classifiers based on superficial features in the presence of adversaries. This kind of attack have been mentioned in Xu et. al.[14], he is capable of automatically producing evasive maldoc variants. Here, for each iteration and for every sample, operation such as addition, deletion and replace to the PDF structure tree is performed via genetic like programming. During the whole process, the malicious behavior of the sample should maintain exactly the same but the ability to confuse/evade the classifier is stronger at each iteration. We call this kind of attack ‘Classifier Evasion Attack’ according to Xu et. al.[14].

An implicit assumption is that structural/behavioral discrepancies exist between benign and malicious documents and such discrepancies can be observed. Since the document must follow a public format specification, commonalities (structural or behavioral) are expected in benign documents. If a document deviates largely from the specification or the common patterns of benign samples, it is more likely to be a malicious document. In other words, a hyper-plane should always be found and posited in a high dimensional feature space to clearly separate the malicious and benign samples. But this assumption doesn’t hold if we can answer the following research questions:

* Can we evade the classifier by adding/deleting/replacing content to the malicious PDF files while still keeping the malicious behavior?
* Can we evade the classifier by gradually adding malicious stuff to the benign PDF files and still fly under the radar without being detected?

The work by Srndic et. al.[4], answering the 1st question, starts from malicious files which we call it ‘Mimicry Attack’; While Maiorca et. al.[10], answering the 2th question, starts from benign files which we call it ‘Reverse Mimicry Attacks’. Both work show that how a maldoc can systematically evades detection.

In summary, for attacking external parser, the current technique is called ‘Parser-Confusion Attacks’; For attacking ML models, the current technique is called ‘Automatic Classifier Evasion Attacks’; For the assumed detectable discrepancy, the current existing attack is called ‘Mimicry and Reverse Mimicry’.

It is no doubts that attacks mentioned above have raised significant challenges on the security of the ML models, extending to the entire framework. Because of this, in this paper, we not only propose sth. that can achieve high accuracy, but a truly secured, robust and working model under our effective defense strategies.