## Related Work

Existing malicious document detection methods can be classified broadly into two categories: dynamic and static analysis. In dynamic analysis, malicious documents are executed and examined in a specially created environment in order to capture the samples’ malicious behavior; in static analysis, the detection is carried out without code execution but with static scanning and examination for the header, binary-level N-gram of files, and others. In general, the advantages of static analysis are ease of deployment and good speed (but relatively low accuracy). Compared to static analysis, dynamic analysis, although suffering from low speed and intense resource consumption, exhibits the highest accuracy. Nowadays, both techniques have already had a large number of success stories. More advanced solutions along this line usually involve the hybrids of dynamic and static detection methods (see Maiorca *et al*. [9] for details). A summary of existing methods is presented in Table1.

Table 1. Taxonomy of malicious PDF document techniques partially based on platform diversity [8] with the addition of works after 2016 as well as summaries parsers, ML, 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 [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 [20] | Tap point | 2016 | N | N | Y |
| Memory | Violation of invariants [19] | CWXDetector | 2012 | N | N | N |
|  | OS | Platform diversity [21] | PlatPal | 2017 | Y | N | Y |

From Table 1, we can conclude that the main focus of static analysis is JavaScript or file metadata. Typical detection techniques include Shellcode and Opcode Sig–based MPScan [13], and structure- and content-based classification [9]. However, dynamic analysis mainly focuses on extracting the JavaScript Snippet from the file and running them directly to enable malicious behavior detection. Typical work in this line includes behavior-based analysis [20] and platform diversity-based analysis [21].

We can also see from Table1 that all but three methods use either open-source or home-grown parsers and assume their capability. However, Carmony *et al*. [20] shows that these parsers are typically incomplete and make oversimplified assumptions in regard to where JavaScript can be embedded. This leads to one of the most important questions in this research: Is the external parser robust or not? This is because the design and implementation of these kinds of external parsers are usually simple without being designed to be secured; in this case, only a little effort is needed for the successful evasion of malicious malware. This kind of attack is called “parser-confusion attacks” by Carmony *et al*. [20].

An important conclusion can be drawn from Table 1, namely that ML, in general, is fit for static rather than dynamic analysis. We are not aware of a paper on dynamic analysis that considers ML, while ML has been the “default standard” in nearly all static analysis papers. Typical ML work here includes PDFRATE [24] and PDF Malware Slayer [11]. Nearly all of the aforementioned works claim that their classifiers can attain high accuracy in resource-intensive environments, but seldom mention the security of their deployed ML models, much less the need for a comprehensive study of adversarial ML. This raises serious doubts about the effectiveness of classifiers based on superficial features in the presence of adversaries. An attack exploiting this vulnerability is mentioned in Xu *et al*. [14], one that can automatically produce evasive maldoc variants. Here, for each iteration and for every sample, operations like addition, deletion, and replace with respect to the PDF structure tree is performed via genetic-programming-like operations. During the entire process, the malicious behavior of the sample should remain exactly the same, but the ability to confuse and evade the classifier is stronger at each iteration. This kind of attack is called a “classifier evasion attack” by 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. However, this assumption does not hold if we can answer the following research questions:

* Can we evade the classifier by adding, deleting, or replacing content of the malicious PDF files while still keeping the malicious behaviors of files?
* Can we still evade the classifier by gradually adding malicious content to the benign PDF?

The work of Srndic *et al*. [4], starting with malicious files, has answered the first question, which refers to what is called a “mimicry attack.” Maiorca *et al*. [10], starting with benign files, has answered the second question, which refers to what is called a “reverse mimicry attack.” Both works show how a malicious document can systematically evade detection.

In summary, for attacking an external parser, the current technique is called a “parser-confusion attack.” For attacking ML models, the current technique is called an “automatic classifier evasion attack.” For the assumed detectable discrepancy, the existing attack is called “Mimicry and Reverse Mimicry.”

There is no doubt that the above-mentioned attacks have raised significant challenges to the security of ML models, extending to the entire framework. Therefore, in this paper, we propose not only some approaches that can achieve high accuracy, but also a truly secure, robust, and working model under our effective defense strategies.